CSE 318

Assignment 3

SOLVING THE MAX CUT PROBLEM BY GRASP

Name : Md. Zulkar Naim

Student Id : 1905016

Max cut is a NP hard problem. In this assignment, we are trying to optimize solution to graph max cut problem using GRASP. The following techniques have been used.

Greedy:

For greedy, the algorithm mentioned is the specification has been used. That is in the first step, we choose the highest weight edge and assign its two endpoints in two different set. And in each step, we add a new vertex to these set from the remaining vertex set greedily on basis of contribution to max cut value.

Semi Greedy:

For semi greedy, the algorithm used in greedy is modified to run as a semi greedy algorithm. Here the value-based method has been implemented. For that a suitable α is chosen and based on that alpha in each step a restricted candidate list is generated. A random vertex is selected from the restricted candidate list.

Local Search:

For local search, in each step the first improving neighbor is chosen. As the neighborhood becomes large for large value of vertex number, hence it is chosen. And local search stops if it fails to find any improving neighbor. That is, it is stuck in an optimum.

GRASP:

For GRASP, the algorithm mentioned in the specification has been used. That is, GRASP is run for a fixed amount of iteration. In each iteration, a semi greedy solution is constructed and then local search is applied on that solution to optimize it. A best solution is maintained over the iterations. After finishing the best solution becomes the solution of the GRASP.

Summary Table:

Problem				Constructive algorithm				Local search		GRASP		Known best solution or upper bound
Name	|V| or n	|E| or m	Weight Range	Simple Randomized	Simple Greedy	Semi Greedy		Simple Local		GRASP-1		
						alpha = 0.6	alpha = 0.8	No. of iterations	Best value	No. of iterations	Best value	
G1	800	19176	[1,1]	9592	11247	11196	11249	99	11393	50	11452	12078
G2	800	19176	[1,1]	9587	11291	11190	11241	102	11390	50	11463	12084
G3	800	19176	[1,1]	9578	11243	11201	11230	97	11385	50	11487	12077
G4	800	19176	[1,1]	9581	11334	11194	11250	98	11398	50	11518	Unknown
G5	800	19176	[1,1]	9609	11294	11203	11235	95	11401	50	11484	Unknown
G6	800	19176	[-1,1]	79	1791	1667	1747	126	1920	50	1991	Unknown
G7	800	19176	[-1,1]	-70	1582	1512	1583	125	1763	50	1840	Unknown
G8	800	19176	[-1,1]	-99	1614	1499	1579	123	1755	50	1864	Unknown
G9	800	19176	[-1,1]	-23	1610	1547	1626	129	1799	50	1891	Unknown
G10	800	19176	[-1,1]	-77	1590	1515	1571	126	1742	50	1826	Unknown
G11	800	1600	[-1,1]	14	480	453	484	9	469	50	496	627
G12	800	1600	[-1,1]	3	482	438	473	9	452	50	476	621
G13	800	1600	[-1,1]	19	498	462	495	10	482	50	506	645
G14	800	4694	[1,1]	2352	2938	2938	2946	23	2970	50	2995	3187
G15	800	4661	[1,1]	2333	2920	2915	2921	23	2949	50	2969	3169
G16	800	4672	[1,1]	2334	2915	2926	2927	25	2952	50	2991	3172
G17	800	4667	[1,1]	2341	2918	2922	2925	25	2948	50	2966	Unknown
G18	800	4694	[-1,1]	23	819	776	826	56	857	50	889	Unknown
G19	800	4661	[-1,1]	-61	764	684	740	54	765	50	807	Unknown
G20	800	4672	[-1,1]	-22	770	728	770	54	801	50	847	Unknown
G21	800	4667	[-1,1]	-26	774	699	755	57	788	50	834	Unknown
G22	2000	19990	[1,1]	9997	12853	12721	12786	124	12923	50	13003	14123
G23	2000	19990	[1,1]	9997	12747	12724	12769	118	12931	50	12998	14129
G24	2000	19990	[1,1]	10006	12760	12725	12783	124	12923	50	12995	14131
G25	2000	19990	[1,1]	9994	12814	12747	12781	120	12934	50	13009	Unknown
G26	2000	19990	[1,1]	9984	12770	12712	12774	126	12929	50	12988	Unknown
G27	2000	19990	[-1,1]	-32	2631	2537	2662	181	2855	50	2949	Unknown
G28	2000	19990	[-1,1]	-57	2643	2516	2633	185	2819	50	2897	Unknown
G29	2000	19990	[-1,1]	32	2663	2604	2732	189	2918	50	3002	Unknown
G30	2000	19990	[-1,1]	39	2712	2585	2735	179	2917	50	3013	Unknown

G31	2000	19990	[-1,1]	-27	2727	2521	2637	187	2837	50	2977	Unknown
G32	2000	4000	[-1,1]	17	1222	1126	1193	20	1164	50	1210	1560
G33	2000	4000	[-1,1]	-11	1198	1101	1183	21	1141	50	1184	1537
G34	2000	4000	[-1,1]	-23	1170	1101	1178	21	1148	50	1194	1541
G35	2000	11778	[1,1]	5883	7377	7367	7385	58	7444	50	7494	8000
G36	2000	11766	[1,1]	5876	7408	7364	7374	57	7434	50	7465	7996
G37	2000	11785	[1,1]	5888	7397	7370	7382	59	7451	50	7476	8009
G38	2000	11779	[1,1]	5898	7376	7365	7385	60	7445	50	7488	Unknown
G39	2000	11778	[-1,1]	12	2026	1789	1950	149	2050	50	2121	Unknown
G40	2000	11766	[-1,1]	-47	1992	1847	1959	136	2041	50	2089	Unknown
G41	2000	11785	[-1,1]	-10	2090	1820	1988	139	2052	50	2125	Unknown
G42	2000	11779	[-1,1]	58	2083	1897	2047	146	2121	50	2198	Unknown
G43	1000	9990	[1,1]	4991	6362	6341	6379	65	6451	50	6492	7027
G44	1000	9990	[1,1]	4998	6384	6342	6375	63	6444	50	6498	7022
G45	1000	9990	[1,1]	4996	6328	6341	6378	65	6444	50	6482	7020
G46	1000	9990	[1,1]	4999	6417	6344	6369	61	6441	50	6497	Unknown
G47	1000	9990	[1,1]	4979	6397	6351	6378	61	6456	50	6521	Unknown
G48	3000	6000	[1,1]	3000	6000	6000	6000	1	6000	50	6000	6000
G49	3000	6000	[1,1]	2997	6000	6000	6000	1	6000	50	6000	6000
G50	3000	6000	[1,1]	2989	5880	5880	5880	1	5880	50	5880	5988
G51	1000	5909	[1,1]	2951	3685	3687	3696	32	3729	50	3752	Unknown
G52	1000	5916	[1,1]	2951	3719	3694	3697	32	3736	50	3759	Unknown
G53	1000	5914	[1,1]	2953	3671	3691	3698	30	3728	50	3749	Unknown
G54	1000	5916	[1,1]	2952	3705	3691	3696	32	3727	50	3752	Unknown

The summary table is generated for all the provided input files.

Explanation:

The problem column briefly describes the problem that is name, node count, edge count, edge weight range etc.

Constructive algorithm column mentions different constructive algorithms. First is simple random, which is just a randomized assignment of vertex. As a result, from the table it can be seen that it doesn't perform that well. Its value is averaged over 50 steps. Next is simple greedy. From the table we can see that among all constructive algorithm it performs the best. As it is pure greedy, it chooses the best possible solution in each step. So, its result matches with our expectation. At last semi greedy algorithm is used for $\alpha = 0.6$ and $\alpha = 0.8$. From the table, we can see

that $\alpha = 0.8$ performs better than $\alpha = 0.6$ as it is less randomized. Both of them are averaged over 20 steps.

In local search column, simple local search properties are shown. In each iteration of GRASP, a local search is performed. So, over a full run of GRASP, no. of iterations of local search and its best value is averaged. In almost all cases, local search gives better result than the constructive algorithms. It is also intuitive as local search uses constructive algorithm result and optimizes from that.

In GRASP column, GRASP iteration and best value is shown. Each of the GRASP is run for 50 iterations. Its best value is better than local search as it runs local search in each iteration and outputs the maximum value among them.

If we compare these values, we can see that they are very close to the known best value. So, it can be concluded that our algorithm runs correctly.